Да се създаде клас поддържащ връзка от тип 1..* между обекти от произволен тип. Да се имплементира показаният интерфейс. Много уникални обекти „source“ сочат към един обект „target“. Интерфейсът позволява да се определят всички „source“ сочещи към даден „target“, както и към кой „target“ сочи даден „source“.

```
/**
 * Introduces the notation of many-to-one relation. This is where the M and O of
 * the type signature comes from.
 *
 * Many unique "source" objects refer to one and only "target" object.
 *
 * The class maintains a connection between the target and all the sources that
 * are referring to it.
 *
 * @author Kiril Mitov
 *
 * @param <M>
 *            the type of the "source" objects.
 * @param <O>
 *            the type of the "target" objects.
 */

public interface ManyToOneRelationInterface<M, O> {

      /**
       * Connects the given source with the given target. If this source was
       * previously connected with another target the old connection is lost.
       *
       * @param source
       * @param target
       * @return
       */
      public abstract boolean connect(M source, O target);

      /**
       * @param source
       * @return <code>true</code> if the relation contains the given source
       */
      public abstract boolean containsSource(M source);

      /**
       * @param target
       * @return <code>true</code> if the relation contains the given target
       */
      public abstract boolean containsTarget(O target);

      /**
       * @param source
       * @return the target with which this source is connected
       */
      public abstract O getTarget(M source);

      /**
       * @param target
       * @return all the targets that are connected with this source or empty
       *         collection if there are no sources connected with this target.
       */
      public abstract Collection<M> getSources(O target);

      /**
       * Removes the connection between this source and the corresponding
```

```
target.
	 * Other sources will still point to the same target.
	 *
	 * The target is removed if this was the only source pointing to it and
	 * {@link #containsTarget(Object)} will return false.
	 *
	 * @param source
	 */
	public abstract void disconnectSource(M source);

	/**
	 * Removes the given target from the relation. All the sources that are
	 * pointing to this target are also removed.
	 *
	 * If you take the "result" of {@link #getSources(target)} and after that
	 * call this method then {@link #containsSource(Object)} will return
	 * <code>false</code> for every object in "result".
	 *
	 * @param target
	 */
	public abstract void disconnect(O target);

}
```

Имплементацията трябва да преминава през следния тест.

```
package org.eclipse.jst.javaee.ejb.model.test;

import static org.junit.Assert.assertEquals;
import static org.junit.Assert.assertFalse;
import static org.junit.Assert.assertNull;
import static org.junit.Assert.assertTrue;

import org.eclipse.jst.javaee.ejb.model.ManyToOneRelationInterface;
import org.junit.Before;
import org.junit.Test;

/**
 * @author Kiril Mitov
 *
 */
public class ManyToOneRelationTest {

	private ManyToOneRelationInterface<String, Integer> fixture;

	private ManyToOneRelationInterface<String, Integer> getFixture() {
		return fixture;
	}

	@Before
	public void setUp() {
		fixture = null;
	}

	@Test
	public void testGetTarget() {
```

```
        getFixture().connect("str1", new Integer(1));
        assertEquals(new Integer(1), getFixture().getTarget("str1"));
}

@Test
public void testGetSources() {
        getFixture().connect("str1", new Integer(1));
        assertTrue(getFixture().getSources(new Integer(1)).contains("str1"));
}

@Test
public void testGetSourcesManySources() {
        getFixture().connect("str1", new Integer(1));
        getFixture().connect("str2", new Integer(2));
        assertTrue(getFixture().getSources(new Integer(1)).contains("str1"));
        assertTrue(getFixture().getSources(new Integer(2)).contains("str2"));
}

@Test
public void testGetTargetManySources() {

        getFixture().connect("str1", new Integer(1));
        getFixture().connect("str2", new Integer(1));
        assertEquals(new Integer(1), getFixture().getTarget("str1"));
        assertEquals(new Integer(1), getFixture().getTarget("str2"));
}

@Test
public void testContainsSource() {

        getFixture().connect("str1", new Integer(1));
        getFixture().connect("str2", new Integer(2));

        assertTrue(getFixture().containsSource("str1"));
        assertTrue(getFixture().containsSource("str2"));
        assertFalse(getFixture().containsSource(null));
}

@Test
public void testConainsTargetTwoSources() {
        getFixture().connect("str1", new Integer(1));
        getFixture().connect("str2", new Integer(2));
        assertEquals(new Integer(1), getFixture().getTarget("str1"));
        assertEquals(new Integer(2), getFixture().getTarget("str2"));
        assertTrue(getFixture().containsTarget(new Integer(1)));
        assertTrue(getFixture().containsTarget(new Integer(2)));
}

@Test
public void testDisconnectSource() {
        getFixture().connect("str1", new Integer(1));
        getFixture().connect("str2", new Integer(1));
```

```
		assertEquals(new Integer(1), getFixture().getTarget("str1"));
		assertEquals(new Integer(1), getFixture().getTarget("str2"));

		getFixture().disconnectSource("str1");
		assertFalse(getFixture().containsSource("str1"));
		assertNull(getFixture().getTarget("str1"));
		assertEquals(new Integer(1), new Integer(getFixture().getSources(new
Integer(1)).size()));
		assertTrue(getFixture().getSources(new Integer(1)).contains("str2"));
	}

	@Test
	public void testDisconnect() {
		getFixture().connect("str1", new Integer(1));
		getFixture().connect("str11", new Integer(1));
		getFixture().connect("str2", new Integer(2));
		assertEquals(new Integer(1), getFixture().getSources(new Integer(2)).size());
		assertEquals(new Integer(2), getFixture().getSources(new Integer(1)).size());

		getFixture().disconnect(new Integer(1));
		assertFalse(getFixture().containsTarget(new Integer(1)));
		assertFalse(getFixture().containsSource("str1"));
		assertFalse(getFixture().containsSource("str11"));
		assertTrue(getFixture().containsSource("str2"));
		assertTrue(getFixture().containsTarget(new Integer(2)));
	}

}
```